Estimating Markov Switching model using Gibbs sampling with a statistical computing software R *

Atsushi Matsumoto †

2008.6.10

Abstract

The objective of this paper is to provide readers with the program to estimate a Markov switching model with time varying transition probability(Filardo, 1994) by using a statistical computing software R. Although many of the previous studies estimating the model have conducted the estimation by the maximum likelihood estimation, this paper utilizes Gibbs sampling method. Using Gibbs sampling method enables us to estimate more complicated models which are impossible or difficult to be estimated by the maximum likelihood estimation.

1 Introduction

In many parts of time series analysis, a Markov switching model(Hamilton, 1989) and a Markov switching model with time varying transition probability(Filardo, 1994) have been used to capture structural changes which cannnot be observed. But these models are highly nonlinear and are difficult to be estimated, or is impossible to be done in some cases by using the maximum likelihood estimation. In such cases, other method should be applied. Following Filardo and Gordon(1998), this paper uses Gibbs sampling method to estimate the model. Using this method enables us to estimate the highly nonlinear model.

This paper is not the first attempt to estimate a Markov switching model by Gibbs sampling. But, according to the author's knowledge, there has been no attempt to do it with a statistical computing software R. The advantage of using R is that we can use it by free of charge, while GAUSS or Matlab is expensive. Therefore, providing the program in R language is a contribution of this paper.

This paper proceeds as follows. The next section reviews the model. Section 3 explains the program used for estimating the model. Section 4 argues the advantages of using Gibbs sampling method.

2 Model

The model of interest is a Markov switching model with time varying transition probability(MS-TVTP model) with a mean-deviation, p^{th} order-autoregressive form:

$$y_t - m(s_t) = \phi_1\big(y_{t-1} - m(s_{t-1})\big) + \cdots + \phi_p\big(y_{t-p} - m(s_{t-p})\big) + e_t,\ e_t \ \sim \text{i.i.d. } \mathcal{N}(0, \sigma^2), \quad (1)$$

where $m(s_t)$, for instance, indicates that the parameter is dependent on the state at t, $s_t = 0, 1$, which cannnot be observed but is assumed to follow a discrete Markov chain such that $\Pr(s_t|s_{t-1}, s_{t-2}, \cdots) = \Pr(s_t|s_{t-1})$. And the value of s_t is determined by a latent variable s_t^* as the following rule:

$$\begin{aligned} s_t = 1 \text{ if } s_t^* &= \boldsymbol{z}_{t-1}'\boldsymbol{\gamma}^{(s_{t-1})} + u_t \geq 0, \\ s_t = 0 \text{ if } s_t^* &= \boldsymbol{z}_{t-1}'\boldsymbol{\gamma}^{(s_{t-1})} + u_t \leq 0, \end{aligned} \quad (2)$$

*The estimation is conducted by using a statistical computing software R, version 2.4.0. Although all the programs are confirmed to be worked precisely, all the remaining errors are mine. In writing this note, I refered the GAUSS program written by Professor Martin Ellison, University of Warwick. I am thankful to him for his program distributed at `http://www2.warwick.ac.uk/fac/soc/economics/staff/faculty/ellison/software/`.

†**E-mail:** atsushi-mail@hcc6.bai.ne.jp

where the value of s_{t-1} is given, $\boldsymbol{z}_{t-1}$ is a vector of exogenous varibales avairable at $t-1$, and u_t is assumed to follow the standard normal distribution, i.e., $\mathrm{E}(u_t)=0$ and $\mathrm{Var}(u_t)=1$. Since u_t follows such a distribution, the transition probabilities from s_{t-1} to s_t for all the possible values of states are given by:

$$\begin{aligned} s_t = 1 \text{ and } s_{t-1} = 1 \quad &\text{with } \Phi(\boldsymbol{z}_{t-1}'\boldsymbol{\gamma}^{(1)}), \\ s_t = 1 \text{ and } s_{t-1} = 0 \quad &\text{with } \Phi(\boldsymbol{z}_{t-1}'\boldsymbol{\gamma}^{(0)}), \end{aligned} \tag{3}$$

where $\Phi()$ is the cumulative distribution function for the standard normal distribution. Note that $s_t=0$ and $s_{t-1}=1$ with the probability $1-\Phi(\boldsymbol{z}_{t-1}'\boldsymbol{\gamma}^{(1)})$, and that $s_t=0$ and $s_{t-1}=0$ with the probability $1-\Phi(\boldsymbol{z}_{t-1}'\boldsymbol{\gamma}^{(0)})$.

Letting $m(s_t) := m_0 + m_1 s_t$ and $\boldsymbol{\phi} := (\phi_1 \cdots \phi_p)'$, the parameters to be estimated are

$$\boldsymbol{\theta} := \left\{m_0,\ m_1,\ \boldsymbol{\phi},\ \sigma^2,\ \boldsymbol{\gamma}^{(0)},\ \boldsymbol{\gamma}^{(1)},\ \{s_t\}_{t=p+1}^{n},\ \{s_t^*\}_{t=p+1}^{n}\right\}, \tag{4}$$

where all the s_t and s_t^* are included here because they are unobseved and n is the number of observation. In Gibbs sampling estimation, unobserved variables(s_t and s_t^*) are regarded as paramteres and can be obtained. For computation, we denote the information set avairable at t by $\boldsymbol{\Omega}_t$. Note that $\boldsymbol{\Omega}_t$ includes $y_1, \cdots, y_t$ and $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_t$. The parameter set is divided as $\boldsymbol{\theta} = (\boldsymbol{\theta}_1 \cdots \boldsymbol{\theta}_5)$ where

$$\begin{aligned} \boldsymbol{\theta}_1 &:= \{\boldsymbol{\phi},\ \sigma^2\}, \\ \boldsymbol{\theta}_2 &:= \{m_0,\ m_1\}, \\ \boldsymbol{\theta}_3 &:= \{s_t\}_{t=p+1}^{n}, \\ \boldsymbol{\theta}_4 &:= \{s_t^*\}_{t=p+1}^{n}, \\ \boldsymbol{\theta}_5 &:= \{\boldsymbol{\gamma}\}. \end{aligned} \tag{5}$$

In the next section, the procedures and programs to obtain these parameters are explained. As an example, this paper utilizes the data for business cycle and the monetary policy in Japan, i.e., the composite coincident index of business condition index(CCI) and the uncollateralized overnight call rate(CALL). In this example, y_t is the growth rate of CCI($y_t = 100 \cdot \Delta \ln \mathrm{CCI}_t$) and $\boldsymbol{z}_t$ is assumed to include CALL_t($\boldsymbol{z}_t = (1\ s_t\ \mathrm{CALL}_t\ \mathrm{CALL}_t s_t)'$). Then the structure of the determination of the state is rewritten as:

$$s_t^* = \gamma_0 + \gamma_1 s_{t-1} + \gamma_2 \mathrm{CALL}_{t-1} + \gamma_3 \mathrm{CALL}_{t-1} s_{t-1} + u_t, \quad s_{t-1} = i \text{ is given.} \tag{6}$$

Including s_{t-1} and the cross term $\mathrm{CALL}_{t-1}s_{t-1}$ is for considering the effect of the monetary policy variable, separating the state. Therefore $\boldsymbol{\gamma}$ becomes $\boldsymbol{\gamma} = (\gamma_0\ \gamma_1\ \gamma_2\ \gamma_3)'$.

3 Estimation Procedure

This section overviews the procedure and the program to estimate an MS-TVTP model by using Gibbs sampler. Gibbs sampler is a class of Markov Chain Monte Carlo(MCMC) which enables us to estimate the model even in the case where the ordinal maximum likelihood estimation cannot be applied. The discussion in this section is entirely based on Albert and Chib(1993), and Filardo and Gordon(1998). See them for the detail. Then we here use Hungarian notation in which the name of a variable indicates its type or intended use: words beginning with `v` represent vectors, words beginning with `m` represent matrices and other words are basically scalar in this note.

The program begins with loading the data for CCI and CALL to construct the corresponding data vectors `vY` and `vMpol`, respectively, and the prior vector for the values of all the s(`vS`). The prior for s_t is the recession data announced by Economic and Social Research Institute(ESRI) in Japan. If ESRI announces that Japanese economy is in recession at t, $s_t=0$. The data of `vY` and `vS` can be obtained from ESRI website, and that of `vMpol` can be obtained from the Bank of Japan website.

```
data <-read.table("h:\\ci.txt",header=F)
data <-data.frame(data)
vY <-data[,2]
vS <-data[,3]
vY <- (log(vY[139:211])-log(vY[138:210]))*100
```

```
vS <- vS[139:211]

data2 <-read.table("h:\\callrate.txt",header=F)
data2 <-data.frame(data2)
vMpol <-data2[,2]
vMpol <- (vMpol[73:145])*0.1
```

In the above, `vMpol` is multiplied by 0.1 in order to conduct an estimation smoothly. Then we set other conditions:

```
lag <-2
iteration <-15000
burnin <-5000
when <-500
t <-length(vY)
```

Now `lag` is set to 2 because we here consider AR(2) model. And `iteration` is the number of drawing samples, `burnin` is the number of burn-in, which is the number of samples to be discarded for eliminating the influence of initial values, and `when` is how often the estimation result is printed. `t` is the number of observations. Next we here define the matrices for saving:

```
mPhi <-matrix(0,nrow=lag,ncol=(iteration-burnin))
mM <-matrix(0,nrow=2,ncol=(iteration-burnin))
mG <-matrix(0,nrow=4,ncol=(iteration-burnin))
mRec <-matrix(0,nrow=t,ncol=(iteration-burnin))
```

The above commands mean that `mPhi` is a matrix for saving $\boldsymbol{\phi}$, `mM` for $\boldsymbol{m}$, `mG` for $\boldsymbol{\gamma}$ and `mRec` for $\{\Pr(s_t = 0|\boldsymbol{\Omega}_t)\}_{t=1}^{n}$. Note that the above matrices(with zeros) have the row equal to the parameters number and the column equal to the number of saved obervations.

3.1 Obtain $\boldsymbol{\theta}_1$

In Gibss sampling, we need to consider the prior and the posterior distribution for parameters. Now, as to $\boldsymbol{\phi}$ and σ^2, we assume that $\boldsymbol{\phi}$ follows the multivariate normal distribution and σ^2 follows the inverse gamma distribution, which indicates that the inverse of σ^2 follows the gamma distribution. Therefore we set the prior as:

$$\begin{aligned} \boldsymbol{\phi} &\sim \mathcal{N}(\boldsymbol{\phi}_0, \sigma^2 \boldsymbol{P}_0), \text{ given } \sigma^2, \\ \sigma^2 &\sim \mathcal{G}^{-1}\Big(\frac{v_0}{2}, \frac{v_0\sigma_0^2}{2}\Big), \end{aligned} \tag{7}$$

where $\mathrm{E}(\sigma^{-2}) = \sigma_0^{-2}$ and $\mathrm{Var}(\sigma^{-2}) = 2/v_0\sigma^4$. The values with subscript 0(say, $\boldsymbol{\phi}_0$) are the prior values which we need to set before Gibbs sampling. In the program, we write:

```
vPhi0 <-rbind(-0.26,0.001)
mP0 <-rbind(cbind(100,0),
            cbind(0,100))
```

Here `vPhi0` and `mP0` correspond to $\boldsymbol{\phi}_0$ and $\boldsymbol{P}_0$, respectively. Now the prior values for `vPhi0` and `mP0` are not limited to the above. Then letting $\tilde{y}_t := y_t - m_0 - m_1 s_t$ given the values of m_0, m_1 and s_t, Eq.(1) can be rewritten as a linear model without latent variables:

$$\tilde{y}_t = \phi_1 \tilde{y}_{t-1} + \cdots + \phi_p \tilde{y}_{t-p} + e_t. \tag{8}$$

Collecting all the LHS variables into $\tilde{\boldsymbol{y}}$, the RHS variables into $\tilde{\boldsymbol{X}}$ and the error terms into $\boldsymbol{e}$ for all t, the above model becomes $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{X}}\boldsymbol{\phi} + \boldsymbol{e}$. Then the posterior distribution of $\boldsymbol{\phi}$ and σ^2 has the normal-gamma form:

$$\begin{aligned} \boldsymbol{\phi} &\sim \mathcal{N}(\boldsymbol{\phi}_1, \sigma^2 \boldsymbol{P}_1), \text{ given } \sigma^2, \\ \sigma^2 &\sim \mathcal{G}^{-1}\Big(\frac{v_1}{2}, \frac{v_1\sigma_1^2}{2}\Big), \end{aligned} \tag{9}$$

where

$$\begin{aligned} \boldsymbol{P}_1 &= (\boldsymbol{P}_0^{-1} + \tilde{\boldsymbol{X}}'\tilde{\boldsymbol{X}})^{-1}, \\ \boldsymbol{\phi}_1 &= \boldsymbol{P}_1(\boldsymbol{P}_0^{-1}\boldsymbol{\phi}_0 + \tilde{\boldsymbol{X}}\tilde{\boldsymbol{y}}), \\ v_1 &= v_0 + (n - p), \\ \sigma_1^2 &= v_1^{-1}\big(v_0\sigma_0^2 + (\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\boldsymbol{\phi}_0)(\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\boldsymbol{\phi}_0)' + (\boldsymbol{\phi}_1 - \boldsymbol{\phi}_0)'\boldsymbol{P}_0(\boldsymbol{\phi}_1 - \boldsymbol{\phi}_0)\big). \end{aligned} \tag{10}$$

In the program, the above generation rule can be written as below. Firstly, `vYt` is defined, which is the vector with typical element $\tilde{y}_t := y_t - m_0 - m_1 s_t$ given the prior for $\boldsymbol{m}$ and s_t for all t. `mXt` is firstly defined as the vector such as `mXt`$= (\tilde{y}_p \cdots \tilde{y}_{t-1})'$, which is the `lag`$^{\text{th}}$ column of $\tilde{\boldsymbol{X}}$.

```
vYt <- vY-cbind(matrix(1,nrow=t,ncol=1),vS)%*%vM0
mXt <-vYt[lag:(t-1)]
```

We then attempt to construct $\tilde{\boldsymbol{X}}$ as follows. The vector `mXt` is transformed into $\tilde{\boldsymbol{X}}$, using `while` function ,and `cbind` function which is an R function to attach column vectors. Consequently, `mXt` has the typical row $(\tilde{y}_{t-1} \cdots \tilde{y}_p)$ and the vector `vYt` is newly defined as `vYt` $= (\tilde{y}_{p+1} \cdots \tilde{y}_n)'$. The `vYt` and `mXt` created as above yield the model $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{X}}\boldsymbol{\phi} + \boldsymbol{e}$.

```
i <-2
while(i <= lag){
       mXt <-cbind(mXt,vYt[(lag+1-i):(t-i)])
       i <-i+1
}
vYt <-vYt[(lag+1):t]
```

In the below, `vPhi1` and `mP1` correspond to $\boldsymbol{\phi}_1$ and $\boldsymbol{P}_1$, respectively. And `vE` is the residual vector with typical element e_t, `v1` is v_1, where v_0 is set to be zero here, and `sig1` corresponds to σ_1^2.

```
vPhi1 <- solve(solve(mP0)+t(mXt)%*%mXt)%*%(solve(mP0)%*%vPhi0+t(mXt)%*%vYt)
mP1 <-solve(solve(mP0)+t(mXt)%*%mXt)
vE <-vYt-mXt%*%vPhi0
v1 <-(t-lag)
sig1 <-(v1)^(-1)*(t(vE)%*%(vE)+t(vPhi1-vPhi0)%*%mP0%*%(vPhi1-vPhi0))
sigg <-(rgamma(1,shape=v1/2,scale=(v1*sig1/2)^(-1))^(-1))
vPhig <-vPhi1+sqrt(sigg)*t(chol(mP1))%*%rnorm(lag)
```

Since the posterior mean and variance of $\boldsymbol{\phi}$ and the posterior degree of freedom for σ^2 have been defined in the above, we can generate $\boldsymbol{\phi}$ and σ^2 from their posterior distribution. The generated $\boldsymbol{\phi}$ from the posterior is `vPhig`, the generated σ^2 from the posterior is `sigg`. And the generated values are used as the next step's prior.

```
vPhi0 <-vPhig
mP0 <-mP1
```

3.2 Obtain $\boldsymbol{\theta}_2$

In the process of drawing $\boldsymbol{\theta}_1$, we let the values of m_0 and m_1 be given. Here we consider drawing them conditional on s_t and $\boldsymbol{\phi}$. Now we set the prior for $\boldsymbol{m} := (m_0 \ m_1)'$ as:

$$\boldsymbol{m} \sim \mathbf{1}_{(m_1>0)}\mathcal{N}(\boldsymbol{m}_0, \sigma^2\boldsymbol{M}_0), \tag{11}$$

where the indicator function $\mathbf{1}_{(m_1>0)}$ is for identifying m_1. This condition means that the mean growth rate of CCI is positive in expansion and is negative in recession. In the program, we firstly write as below to set the prior for $\boldsymbol{m}$:

```
vM0 <-rbind(-1.2,1.5)
mM0 <-rbind(cbind(500,0),
            cbind(0,500))
```

where `vM0` and `mM0` correspond to $\boldsymbol{m}_0$ and $\boldsymbol{M}_0$, respectively. Of course, the above values are not restrictive. But you must set the first element of `vM0`(here, -1.2) to be negative and its second one to be positive(here, 1.5) for identification. Conditional on s_t and $\boldsymbol{\phi}$, we define $\ddot{y}_t := y_t - \phi_1 y_{t-1} - \cdots - \phi_p y_{t-p}$, $\ddot{s}_t := s_t - \phi_1 s_{t-1} - \cdots - \phi_p s_{t-p}$ and $\ddot{\phi} := 1 - \phi_1 - \cdots - \phi_p$ to rewrite Eq.(1) as:

$$\ddot{y}_t = m_0\ddot{\phi} + m_1\ddot{s}_t + e_t. \tag{12}$$

Collecting all the LHS variables into $\ddot{\boldsymbol{y}}$ and all the RHS variables($\ddot{\phi}$ and $\ddot{s}_t$) into $\ddot{\boldsymbol{X}}$ for all t, the posterior distribution of $\boldsymbol{m}$ has the truncated normal form:

$$\boldsymbol{m} \sim \mathbf{1}_{(m_1>0)}\mathcal{N}(\boldsymbol{m}_1, \sigma^2\boldsymbol{M}_1), \tag{13}$$

where

$$\begin{aligned} \boldsymbol{M}_1 &= (\boldsymbol{M}_0 + \ddot{\boldsymbol{X}}'\ddot{\boldsymbol{X}})^{-1}, \\ \boldsymbol{m}_1 &= \boldsymbol{M}_1(\boldsymbol{M}_0^{-1}\boldsymbol{m}_0 + \ddot{\boldsymbol{X}}'\ddot{\boldsymbol{y}}). \end{aligned} \tag{14}$$

In the program, we firstly attempt to construct the vector $\ddot{\boldsymbol{y}}$ and $\ddot{\boldsymbol{X}}$. In order to do so, we define the vector `mXd`, `mS` and `mD` which, at this point, have the typical element y_t, s_t and 1, respectively.

```
mXd <-vY[lag:(t-1)]
mS <-vS[lag:(t-1)]
mD <-matrix(1,nrow=(t-lag),ncol=1)
```

Then, using a `while` function, `mXd` and `mS` are trasformed into the matrices which have the typical row $(y_t \cdots y_p)$ and $(s_t \cdots s_p)$, respecively. `mD` is also transformed into the $(n-p) \times p$ matrix with all the elements being one.

```
i=2
while( i<=lag){
       mXd <-cbind(mXd,vY[(lag+1-i):(t-i)])
       mS <-cbind(mS,vS[(lag+1-i):(t-i)])
       mD <-cbind(mD,matrix(1,nrow=(t-lag),ncol=1))
       i <-i+1
}
```

After that, `mS`, `mD` and `vYd` are transformed into the vector with typical element $\ddot{s}_t$, $\ddot{\phi}$ and $\ddot{y}_t$. `mXd` consequently corresponds to $\ddot{\boldsymbol{X}}$.

```
mS <-vS[(lag+1):t]-mS%*%vPhig
mD <-1-mD%*%vPhig
vYd <-vY[(lag+1):t]-mXd%*%vPhig
mXd <-cbind(mD,mS)
```

Since the prior for $\boldsymbol{m}$ and $\boldsymbol{M}$ have been set, we can set the posterior mean and variance of $\boldsymbol{m}$, which are denoted by `vM1` and `mM1`, respectively. After setting the posterior, the value of $\boldsymbol{m}$ is generated from the posterior distribution, which is `vMg`. But note that, using a `while` function, `vMg` is kept being generated until the second element of `vMg`, m_1, has the positive value.

```
vM1<-solve(solve(mM0)+t(mXd)%*%mXd)%*%(solve(mM0)%*%vM0+t(mXd)%*%vYd)
mM1 <-solve(solve(mM0)+t(mXd)%*%mXd)

vMg <-matrix(0,nrow=2,ncol=1)
while( vMg[2]<=0 ){
       vMg <-vM1+sqrt(sigg)*t(chol(mM1))%*%rnorm(2)
}
vM0 <- vMg
```

The last line means that the generated values for $\boldsymbol{m}$, `vMG`, are used for the next step's prior.

3.3 Obtain $\boldsymbol{\theta}_3$

The procedure for drawing s_t is based on Albert and Chib(1993). Since their procedure is for the time fixed transition probability model, it is modified to fit the time varying transition probability. Let $\boldsymbol{y}_t := \{y_1 \cdots y_t\}$, $\boldsymbol{s}_t := \{s_1 \cdots s_t\}$ and $\boldsymbol{s}_{-t} := \boldsymbol{s}_n \setminus s_t$ where n is the number of all the observations. Then the full conditional distribution for s_t is given by:

$$\begin{aligned} &\Pr(s_t|\boldsymbol{\Omega}_n, \boldsymbol{s}_{-t}) \propto \Pr(s_t|s_{t-1}, \boldsymbol{z}_{t-1}) \Pr(s_{t+1}|s_t, \boldsymbol{z}_t) \Pr(y_t, \cdots, y_p|\boldsymbol{y}_{t-1}, \boldsymbol{s}_p) \prod_{k=p+1}^{t+p} f(y_k|\boldsymbol{y}_{k-1}, \boldsymbol{s}_k), \text{ for } t \le p, \\ &\Pr(s_t|\boldsymbol{\Omega}_n, \boldsymbol{s}_{-t}) \propto \Pr(s_t|s_{t-1}, \boldsymbol{z}_{t-1}) \Pr(s_{t+1}|s_t, \boldsymbol{z}_t) \prod_{k=t}^{t+p} f(y_k|\boldsymbol{y}_{k-1}, \boldsymbol{s}_k), \quad \text{for } p+1 \le t \le n-p+1, \\ &\Pr(s_t|\boldsymbol{\Omega}_n, \boldsymbol{s}_{-t}) \propto \Pr(s_t|s_{t-1}, \boldsymbol{z}_{t-1}) \Pr(s_{t+1}|s_t, \boldsymbol{z}_t) \prod_{k=t}^{n} f(y_k|\boldsymbol{y}_{k-1}, \boldsymbol{s}_k), \quad \text{for } n-p \le t \le n. \end{aligned} \tag{15}$$

We need to calculate these distributions for all the possible values of s_{t-1}, s_t and s_{t+1}. Working backwards from $t = n$, values for s_t can be simulated from a distribution using the probabilities generated by Eq.(15). Firstly, we define the vectors for saving the filtered probabilities, $\Pr(s_t = 0|\boldsymbol{\Omega}_t)$ and $\Pr(s_t = 1|\boldsymbol{\Omega}_t)$, which are denoted by `vQ` and `vP`, respectively.

```
vQ <-matrix(0,nrow=1,ncol=t)
vP <-matrix(0,nrow=1,ncol=t)
```

Then we obtain the probabilites for drawing s_t in backward way, that is, we calculate the probabilites obtained by Eq.(15). Since, for $s_t = 0, 1$, the product of transition probabilities(the first two terms in Eq.(15)) can be written as:

$$\Pr(s_t = 0|s_{t-1}) \Pr(s_{t+1}|s_t = 0) = \Pr(s_t = 0|s_{t-1})\big\{ \Pr(s_{t+1} = 1|s_t = 0)s_{t+1} + \Pr(s_{t+1} = 0|s_t = 1)(1 - s_{t+1})\big\},$$
$$\Pr(s_t = 1|s_{t-1}) \Pr(s_{t+1}|s_t = 1) = \Pr(s_t = 1|s_{t-1})\big\{ \Pr(s_{t+1} = 1|s_t = 1)s_{t+1} + \Pr(s_{t+1} = 0|s_t = 1)(1 - s_{t+1})\big\},$$

we write:

```
i=t
while( i>=(lag+1)){
       latent <-vG0[1]+vG0[2]*vMpol[(i-1)]+vG0[3]*vS[(i-1)]*vMpol[(i-lag)]+vG0[4]*vS[(i-1)]
       vQ[i] <-1-pnorm(latent)
       vP[i] <-pnorm(latent)
```

The above `q[i]` and `p[i]` correspond to the transition probabilities $\Pr(s_t = 0|s_{t-1})$ and $\Pr(s_t = 1|s_{t-1})$ given s_{t-1}, respectively. Then we next consider $\Pr(s_{t+1}|s_t)$ for $s_t = 0, 1$ and $s_{t+1} = 0, 1$.

```
     if (i<t) {latent <-vG0[1]+vG0[2]*vMpol[(i-1)]
 vQ[i] <-vQ[i]*(1-pnorm(latent))*(1-vS[(i+1)])+vQ[i]*pnorm(latent)*vS[(i+1)]
 vP[i] <-vP[i]*(1-pnorm(latent+vG0[3]*vMpol[(i-1)]+vG0[4]))*(1-vS[(i+1)])
+vP[i]*(pnorm(latent+vG0[3]*vMpol[(i-1)]+vG0[4]))*vS[(i+1)]
                }
```

The newly defined `q[i]` and `p[i]` correspond to $\Pr(s_t = 0|s_{t-1}) \Pr(s_{t+1}|s_t = 0)$ and $\Pr(s_t = 1|s_{t-1}) \Pr(s_{t+1}|s_t = 1)$, respectively. Then we next consider the part beggining with $\prod_{k=t}$ in Eq.(15). To calculate this part, we firstly need to obtain e_t in Eq.(1). Write:

```
      if (i>(t-lag)){ m = t-i }
       else m <-lag

       vS0 <-vS
       vS1 <-vS
       vS0[i] <-0
       vS1[i] <-1
       c <-0
       while( c<=m){
              z0 <-vY[(i+c)]-t(vMg)%*%rbind(1,vS0[(i+c)])
              z1 <-vY[(i+c)]-t(vMg)%*%rbind(1,vS1[(i+c)])
```

The above `z0` and `z1` correspond to $y_t - m_0$ and $y_t - m_0 - m_1$ given the values of s_t, m_0 and m_1 generated in the previous subsection. Since we here consider AR model, we also need to calculate $y_{t-1} - m_0 - m_1 s_{t-1}, \cdots, y_{t-p} - m_0 - m_1 s_{t-p}$. Then we write:

```
         j=1
         while( j <=lag ){
         z0 <-z0-t(vPhig[j])%*%(vY[(i+c-j)]-t(vMg)%*%rbind(1,vS0[(i+c-j)]))
         z1 <-z1-t(vPhig[j])%*%(vY[(i+c-j)]-t(vMg)%*%rbind(1,vS1[(i+c-j)]))
            j<-j+1
            }
            vQ[i] <-vQ[i]*exp(-0.5*t(z0)%*%solve(sigg)%*%(z0))
            vP[i] <-vP[i]*exp(-0.5*t(z1)%*%solve(sigg)%*%(z1))
            c <-c+1
     }
```

The newly obtained `z0` and `z1` correspond to $e_t = y_t - m_0 - \phi_1(y_{t-1} - m_0) - \cdots - \phi_p(y_{t-p} - m_0)$ and $e_t = y_t - m_0 - m_1 - \phi_1(y_{t-1} - m_0 - m_1) - \cdots - \phi_p(y_{t-p} - m_0 - m_1)$, respectively. And, using a `while` function, the part beggining with $\prod_{t=k}$ is obtained, which is newely defined as `vQ[i]` and `vP[i]`.

```
        vQ[i] <-vQ[i]/(vQ[i]+vP[i])
        vP[i] <-1-vQ[i]
        r <-runif(1)
        if (r<vQ[i]){ vS[i]=0 } else vS[i]=1
i <-(i-1)
}
```

We set the state as $s_t = 0$ if a random vairable following the uniform distribution in $[0, 1]$ is smaller than $\Pr(s_t = 0|\boldsymbol{\Omega}_t)$ and as $s_t = 1$ otherwise. The uniform random variable in $[0, 1]$ is generated from a function `runif`.

```
if (k>skip){
        mRec[,(k-skip)] <-t(vQ)
        mPhi[,(k-skip)] <-vPhig
        mM[,(k-skip)] <-vMg
}
```

3.4 Obtain $\boldsymbol{\theta}_4$

Since $\{s_t\}_{t=p+1}^n$ is obtained in the previous subsection, it is easy to generate s_t^* by using Eq.(6). But we need to note that any value of s_t^* will not always do. Since $s_t = 1$ if and only if $s_t^* \geq 0$, we need to use a truncated standard normal distribution. Therefore, the procedure for drawing s_t^* is as follows: (i)Generate u_t from a standard normal distribution, (ii)check the value of s_t^* and s_t, (iii)if $s_t^* \geq 0$ when $s_t = 1$, the generated s_t^* is accepted, but if $s_t^* \leq 0$ when $s_t = 1$, repeat (i) and (ii) until a non-negative value of s_t^* is generated. Firstly we define the vector `vSstar` for saving s_t^* for all t.

```
vSstar <-matrix(0,nrow=1,ncol=t)
```

Then, using the obatained s_t in the above subsection, s_t^* is generated from the following program.

```
i <-(lag+1)
while (i <= t){
        sstar <-vG0[1]+vG0[2]*vMpol[(i-1)]+vG0[3]*vS[(i-1)]*vMpol[(i-1)]+vG0[4]*vS[(i-1)]
        r <-rnorm(1)
        if (vS[i]==0){
                while((sstar+r) >=0){
                        r <-rnorm(1)
                }
        } else
                while((sstar+r) <=0){
                        r <-rnorm(1)
                }
vSstar[i] <-sstar+r
i<-i+1
}
```

In the above, s_t^* is obtained by calculating `sstar + r`, where `r` is the standard normal random variable generated by a function `rnorm`. We here use an `if` function to satisfy the condition that $s_t^* \geq 0$ iff $s_t = 1$ and that $s_t^* \leq 0$ iff $s_t = 0$.

3.5 Obtain $\boldsymbol{\theta}_5$

Since we have obtained $\{s_t^*\}_{t=p+1}^n$ in the previous subsection, Eq.(6) becomes a linear regression without latent variables. Hence $\boldsymbol{\gamma}$ can be regarded as the linear regression coefficient. In Eq.(6), collecting all the LHS varibales(s_t^*) into $\boldsymbol{s}^*$, all the RHS variables into $\boldsymbol{W}$ and u_t into $\boldsymbol{u}$ for all t, it can be rewritten as $\boldsymbol{s}^* = \boldsymbol{W}\boldsymbol{\gamma} + \boldsymbol{u}$. Set the prior for $\boldsymbol{\gamma}$:

$$\boldsymbol{\gamma} \sim \mathcal{N}(\boldsymbol{\gamma}_0, \boldsymbol{G}_0). \tag{16}$$

Then its posterior distribution is given by:

$$\boldsymbol{\gamma} \sim \mathcal{N}(\boldsymbol{\gamma}_1, \boldsymbol{G}_1), \tag{17}$$

where

$$\begin{aligned} \boldsymbol{G}_1 &= (\boldsymbol{G}_0 + \boldsymbol{W}'\boldsymbol{W})^{-1}, \\ \boldsymbol{\gamma}_1 &= \boldsymbol{G}_1^{-1}(\boldsymbol{G}_0^{-1}\boldsymbol{\gamma}_0 + \boldsymbol{W}'\boldsymbol{s}^*). \end{aligned} \tag{18}$$

The above can be easiliy written in the program because the above model is a simple linear regression. We here write as below to set the prior for $\boldsymbol{\gamma}$ and $\boldsymbol{G}$:

```
vG0 <-matrix(0,nrow=4,ncol=1)
vG0[1] <-qnorm(1-0.7)
vG0[2] <-0.5
vG0[3] <-0.6
vG0[4] <-qnorm(0.95)-vG0[1]
mG0 <-rbind(cbind(100,0,0,0),
            cbind(0,100,0,0),
            cbind(0,0,100,0),
            cbind(0,0,0,100))
```

Here `vG0` and `mG0` correspond to $\boldsymbol{\gamma}_0$ and $\boldsymbol{G}_0$, respectively. Next we construct the matrix $\boldsymbol{W}$.

```
mW<-cbind(matrix(1,nrow=(t-lag),ncol=1),vMpol[(lag):(t-1)],
          vMpol[(lag):(t-1)]*vS[(lag):(t-1)],vS[(lag):(t-1)])
```

Then, since we have set the prior for $\boldsymbol{\gamma}$ and $\boldsymbol{G}$, after setting the posterior information, the value of $\boldsymbol{\gamma}$ is generated from the posterior distribution. The last line indicates that the generated values for $\boldsymbol{\gamma}$ are used for the next step's prior.

```
mG1 <-solve(solve(mG0)+t(mW)%*%(mW))
vG1 <-mG1%*%(solve(mG0)%*%vG0+t(mW)%*%(vSstar[(lag+1):t]))
vGg <-vG1+t(chol(mG1))%*%rnorm(4)
vG0 <-vGg
mG0 <-mG1
```

The following is for showing the intermediate result only when `k` is a multipls of `when`. This condition is `if ((k%%when)==0)`. For the details of a function `cat` and `par`, see some R manuals by yourself, because they are not focus of this note.

```
cat("Pass no. \n",k)
if (k>burnin){
       gs[,(k-burnin)] <-g
       if ((k%%when)==0){
           cat("********************************************** \n")
           cat("Estimation Result \n")
           cat("********************************************** \n")
           cat("m0 mean \n",mean(mM[1,1:(k-burnin)]))
           cat(" \n")
           cat("m0 std.dev \n",sd(mM[1,1:(k-burnin)]))
           cat(" \n")
                              .
                              .
           cat("gamma4 mean \n",mean(mG[4,1:(k-burnin)]))
           cat(" \n")
           cat("gamma4 std.dev \n",sd(mG[4,1:(k-burnin)]))
           cat(" \n")
           cat("********************************************** \n")
             par(mfrow=c(2,1))
             plot(ts(vS[2:t]),main="Announced Recession")
             plot(ts(mRec[2:t,(k-skip)]),main="Filtered Prob.")
        }
}
k <-(k+1)
}
```

4 Remarks

Using Gibbs sampling method opens a way to estimate highly complicated models. One difference between the maximum likelihood and MCMC is the dealing of latent variables: a latent variable is treated as unobserved variable in the maximum likelihood estimation, while it is treated as parameters in MCMC estimation. This difference in treating latent variables enables us to make the estimated model be easy to be estimated, compared with the maximum likelihood estimation.

Another advantage of MCMC estimation is the problem of degree of freedom. In the maximum likelihood estimation, if the parameters number is large compared with the obervation, the model becomes unstable and the obtained result would not be reliable. But we do not need to worry about it if we use MCMC estimation. In this note, for example, the number of parameteres to be estimated is larger than n, which is the observation number.

Other model modification and program modification should be done, following this note. I lastly write the references used for writing this note.

Albert, J. and Chib, S.(1993) Bayes inference via Gibbs sampling of autoregressive time series subject to Markov mean and variance shifts, *Journal of Business and Economic Statistics*, 11, pp.1-15

Filardo, A.J(1994) Business cycle phases and their transitional dynamics, *Journal of Business and Economic Statistics*, 12, pp.299-308

Filardo, A.J and Gordon, S.F(1998) Business cycle durations, *Journal of Econometrics*, 85, pp.99-123

Hamilton, J.D(1989) A new approach to the economic analysis of nonstationary time series and the business cycle, *Econometrica*, 57, pp.357-384